برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۶

مدرس: رامتین خسروی

طراحان: محمدعرفان دانایی، پارسا دقیق، امیرعلی دهقانی، امیرحسین عارف‌زاده، بهاره عین‌الهی، عرفان فلاحتی، طاها مجلسی، مهدیس میرزائی، امیررضا نادی، پریسا یحیی‌پور

مهلت تحویل: دوشنبه ۵ خرداد ۱۴۰۴، ساعت ۲۳:۵۹

مقدمه

این پروژه به جمع‌بندی آموخته‌های شما در این درس می‌پردازد. انتظار می‌رود مهارت‌هایی را که در تمرین‌های پیشین و سایر بخش‌های درس آموخته‌اید، در پیاده‌سازی این پروژه به کار گیرید. در این تمرین شما باید یک سامانه مدیریت کارهای شخصی پیاده‌سازی کنید. اضافه کردن رویداد و تسک و گزارش‌های متنوع از قابلیت‌های این سامانه هستند. در فازهای بعدی این سامانه به تدریج گسترش می‌یابد. بنابراین زمان کافی به طراحی ساختار این سامانه اختصاص دهید.

نکته قابل توجه در این پروژه این است که بهتر است پروژه به صورت incremental (تدریجی) پیاده‌سازی و آزموده شود. به طوری که ابتدا یک ساختار کلی از پروژه پیاده‌سازی شود و سپس دستورات مختلف مرحله به مرحله به آن اضافه گردد.

شرح تمرین

در این تمرین سامانه مدیریت کارهای شخصی را پیاده‌سازی می‌کنید. این سامانه به کاربران امکان می‌دهد تا رویداد [1]یا تسک ایجاد کنند. رویداد کاری است که باید در بازه زمانی مشخص انجام شود. مانند کلاس‌های دانشگاه. تسک کاری است که انجام دادن آن به بازه خاصی اختصاص ندارد. اما باید تا موعد آن انجام شود. مانند تکالیف هر درس. در صورتی که رویداد جدید با رویداد دیگری تداخل نداشته باشد و در روز تعطیل رسمی نباشد، به برنامه شخصی فرد اضافه می‌شود. اما برای تسک تداخل بررسی نمی‌شود و در هر تاریخ و ساعتی می‌توان تسک ایجاد کرد. رویدادها انواع مختلفی دارند. بعضی از آنها معمولی هستند ولی برخی از آنها در بازه‌های مشخص تکرار می‌شوند. مثلا کلاس درس برنامه‌سازی پیشرفته هر هفته روزهای یکشنبه و سه‌شنبه در ساعت مشخص تکرار می‌شود. بنابراین یک رویداد دوره‌ای از نوع هفتگی است. اما امتحان پایان‌ترم رویدادی دو ساعته است که تاریخ مشخصی دارد و تکرار نمی‌شود. بنابراین رویداد معمولی است. کاربران برای اینکه بتوانند برنامه شخصی خود را مشاهده کنند، از قابلیت گزارش‌گیری سامانه استفاده می‌کنند. کاربر با توجه به نیاز خود می‌تواند از انواع گزارش‌ها استفاده کند. برای مثال می‌تواند تمام کارهای خود را در بازه زمانی مشخص مشاهده کند یا گزارش را محدود به نوع خاصی از کارها کند. برای مثال می‌تواند فقط تسک‌های خود را مشاهده کند.

قالب فایل‌های ورودی

در ابتدای اجرای برنامه، شما باید اطلاعاتی را که در قالب فایل‌های CSV[2] به شما داده می‌شود بخوانید و در برنامه خود به شکل مناسب ذخیره کنید. در ادامه به توضیح جزئیات فایل می‌پردازیم. توجه کنید تضمین می‌شود تمام فایل‌های ورودی درست هستند و مشکلی در آنها وجود ندارد.

پیکربندی نحوه اجرای برنامه
`./UTrello </path/to/holiday/file>`

فایل تعطیلات

این فایل شامل سه ستون روز، ماه، سال است که تعطیلات رسمی تقویم را مشخص می‌کند. تاریخ‌هایی که در این فایل قرار گرفته‌اند، ترتیب خاصی ندارند.

[1] Event

[2] Comma-Separated Values

نمونه فایل تعطیلات

```
day,month,year
1,1,1404
13,1,1404
```

ساعت

در این سامانه برای مشخص کردن زمان فقط از ساعت استفاده می‌شود. اعداد مجاز برای ساعت حسابی هستند و در بازه ۰ تا ۲۳ قرار دارند. در هر دستوری که آرگومان مربوط به ساعت دارد، در صورتی که مقدار وارد شده خارج از بازه ذکر شده باشد، خطای Bad Request چاپ کنید.

تقویم

قالب تاریخ در کل برنامه به شکل YYYY/MM/DD می‌باشد و تضمین می‌شود همواره به درستی ورودی داده می‌شود. تاریخ فعلی برنامه را به طور پیش‌فرض ۱ فروردین ۱۴۰۴ که جمعه است در نظر بگیرید. توجه داشته باشید تقویم برنامه انتها ندارد. برای محاسبه اینکه هر تاریخ چه روزی از هفته است، از الگوریتم زیر استفاده کنید. تضمین می‌شود قالب تاریخ ورودی در تمام دستوراتی که آرگومان تاریخ دارند، به درستی وارد می‌شوند.

محاسبه روز هفته

در دبیرستان با مفهوم هم‌نهشتی آشنا شدیم. یکی از کاربردهای هم‌نهشتی، محاسبه روز هفته بر اساس تاریخ است. برای این کار باید روزهای هفته را کدگذاری کنیم. روز هفته تاریخ مبدا دارای کد ۰ است و سایر روزهای هفته به ترتیب تا ۶ مشخص می‌شوند.

برای مثال ۱ فروردین ۱۴۰۴ جمعه است. کدگذاری روزهای هفته به شکل زیر خواهد بود:

شنبه	یکشنبه	دوشنبه	سه‌شنبه	چهارشنبه	پنج‌شنبه	جمعه
۱	۲	۳	۴	۵	۶	۰

سپس تعداد روزهای بین تاریخ مبدا و تاریخ مقصد را محاسبه می‌کنیم. در این تمرین فرض کنید تمام ماه‌ها ۳۰ روزه هستند و هر سال ۳۶۰ روز است. در نهایت باقی‌مانده این عدد را به ۷ محاسبه می‌کنیم و با کدگذاری بالا مطابقت می‌دهیم.

مثال

می‌خواهیم محاسبه کنیم ۴ تیر ۱۴۰۵ چه روزی از هفته است. تاریخ مبدا سامانه ما همانطور که اشاره شد، ۱ فروردین ۱۴۰۴ و جمعه است.

$$(۲۹ + ۱۴*۳۰ + ۴) \% ۷ = ۵$$

طبق جدول بالا این تاریخ چهارشنبه است. ۲۹ روز مربوط به فاصله تاریخ مبدا (۱ فروردین ۱۴۰۴) تا انتهای همان ماه است. ۱۴ ماه کامل تا آخر خرداد ۱۴۰۵ وجود دارد. در نهایت هم روز ۴ام تیر مدنظر است.

انواع دستورات

در این فاز، منطق برنامه در قالب تعدادی دستور که در ادامه توضیح داده شده است پیاده‌سازی می‌شود. روند استفاده از برنامه به این شکل است که کاربر در برنامه شما با استفاده از رابط خط فرمان[3]، دستوری همراه با آرگومان‌های لازم برای اجرای آن در ورودی استاندارد[4] وارد می‌کند. به عنوان مثال، برای گرفتن یک لیست از اطلاعات موجودیت‌ها از دستور GET و در صورت گرفتن یک عنصر خاص از موجودیت مورد نظر بعد از دستور، شناسه[5] موجودیتی که مایل به گرفتن اطلاعات آن است را نیز وارد می‌کند تا دستور مورد نظرش اجرا شود.

آرگومان‌های هر دستور، پس از علامت ? در دستور می‌آیند و نیز ترتیب خاصی برای آن‌ها وجود ندارد؛ به این معنا که لزومی ندارد آرگومان‌ها به همان ترتیبی که در توضیحات هر دستور گفته شده، وارد شوند. توجه کنید که برای دستوراتی که آرگومان نداریم نیز علامت ? می‌آید.

به عنوان مثال، دستور زیر یک رویداد جدید در سیستم اضافه می‌کند (توضیحات دقیق این دستور بعداً در جای خود ذکر خواهد شد):

نمونه دستور از دسته POST

```
POST event ? date 1404/12/02 start_time 9 duration 1 title "Quiz"
```

نکته دیگری که برای دستورات وجود دارد این است که در ابتدای هر دستور، عبارت GET یا POST یا DELETE یا PUT وارد می‌شود که به این شکل دستورات با توجه به نوع کاری که انجام می‌دهند دسته‌بندی می‌شوند. دستوراتی که برای دریافت اطلاعات از سامانه استفاده می‌شوند در دسته GET قرار می‌گیرند.

این نام‌گذاری دستورها در فازهای آتی پروژه که برنامه خود را روی وب عرضه خواهید کرد معنای خاص پیدا خواهند کرد. همچنین دقت کنید که ممکن است دو دستور با نام‌های مشابه وجود داشته باشند اما در دسته‌های متفاوتی قرار بگیرند، در این صورت ماهیت این دو دستور متفاوت بوده و در صورت فراخوانی آن‌ها، نتایج متفاوتی را مشاهده خواهیم کرد.

---

[3] Command line

[4] Standard input

[5] ID

همینطور دقت داشته باشید که تمامی دستورها پس از اجرا شدن دارای خروجی مشخص هستند که منحصرا ذکر می‌شود. اگر در دستورات وارد شده کاربر، خطایی وجود داشته باشد، چه در دستورات چه در آرگومان‌ها، باید با توجه به توضیحاتی که همراه با هر دستور آمده است، خطای آن را خروجی دهید.
خروجی پروژه شما به صورت خودکار آزموده می‌شود؛ بنابراین خروجی شما باید دقیقاً همانند خروجی خواسته شده باشد. در غیر این صورت نمره‌ی بخش آزمون را از دست خواهید داد.

پاسخ دستورات

به ازای هر دستوری که اقدام به اجرای آن می‌کنیم، پاسخی از سمت سامانه دریافت می‌کنیم. این پاسخ می‌تواند اطلاعاتی که از سامانه خواسته شده است، باشد. اما گونه‌های دیگری از پاسخ نیز وجود دارد که در ادامه توضیح داده خواهد شد (رنگ‌های نمونه صرفا برای خوانایی می‌باشد و نباید از آن‌ها در خروجی استفاده کنید).

پاسخ درخواست موفقیت‌آمیز

اگر دستوری که کاربر وارد می‌کند به درستی انجام شود و به اتمام برسد، این پاسخ نمایش داده می‌شود (در برخی از حالات ممکن است خود دستور خروجی مفصل‌تری داشته باشد که در این صورت این پاسخ نمایش داده نمی‌شود، این حالات در ادامه و در توضیح هر بخش توضیح داده شده‌اند).

خروجی
OK

پاسخ خالی بودن

در صورتی که لیست درخواست‌شده از سامانه هیچ مورد قابل نمایشی نداشته باشد، این پاسخ به کاربر نمایش داده خواهد شد.

خروجی
Empty

پاسخ عدم وجود

در صورتی که دستور وارد شده در لیست دستورات PUT، DELETE، POST، GET وجود نداشت، این پیغام نمایش داده می‌شود. همچنین در صورتی که شناسه وجود نداشته باشد و به طور کلی در هر قسمت که جستجویی انجام می‌شود اما نتیجه‌ای در بر ندارد، این پاسخ داده می‌شود.

خروجی
`Not Found`

پاسخ درخواست اشتباه

اگر اولین قسمت ورودی کاربر، هیچ کدام از دستورهای PUT، POST، GET و DELETE نباشد، این پاسخ نمایش داده می‌شود. همچنین اگر دستور واردشده، اطلاعات کافی برای اجرا را در خود نداشته باشد و یا قالب دستور واردشده، با هیچ کدام از دستوراتی که در ادامه می‌آیند مطابقت نداشته باشد (آرگومان‌های دستور به درستی داده نشده باشند)، این پاسخ نمایش داده می‌شود. توجه داشته باشید که اگر مقادیر آرگومان‌های مورد انتظار درست بود اما تعدادی آرگومان بیشتر نیز داشتیم خطایی دریافت نمی‌شود.

خروجی
`Bad Request`

پاسخ عدم دسترسی (دسترسی غیرمجاز)

اگر یک کاربر دستوری وارد کرد که اجازه دسترسی به آن وجود نداشت این پیغام نمایش داده می‌شود.

خروجی
Permission Denied

سنجش خطاها

اولیت‌بندی سنجش خطاها در اجرای برنامه به صورت زیر می‌باشد:

1. ابتدا بررسی می‌شود که دستور با یکی از متدهای PUT، GET،POST یا DELETE شروع می‌شود. در صورتی که در ابتدای دستور وارد شده یکی از این چهار کلمه نباشد خطای Bad Request نمایش داده می‌شود.
2. پس از آن بررسی می‌شود که دستور وارد شده در لیست دستورات وجود دارد یا خیر؛ برای مثال دستور `GET something_non_existant` در دستورات برنامه نیست. در این حالت باید پاسخ Not Found نمایش داده شود.
3. سپس برای هر دستور اجازه دسترسی بررسی شود که در صورت عدم دسترسی با دستور Permission Denied مواجه شوند. در صورتی که کاربر هنوز لاگین نکرده باشد، یا در صورتی که یک کاربر دستورات یک نوع کاربر دیگر را وارد کند، از مصادیق این خطا هستند.
4. پس از آن حالات خاص هر دستور بررسی می‌شود، تضمین می‌شود در این حالت صرفا با یکی از حالات خاص مواجه هستیم (چند خطا در اینجا رخ نمی‌دهد).

دستورات

در این بخش به توضیح دستورات برنامه می‌پردازیم. توجه داشته باشید برنامه شما نباید به تعداد کاراکتر فاصله[6] در هر دستور حساس باشد و برای آن خطا در نظر بگیرد. این موضوع در تست‌ها نیز بررسی خواهد شد.

ثبت‌نام

این دستور به کاربران امکان می‌دهد تا در سامانه ثبت‌نام کنند. هر فرد با استفاده از یک نام کاربری یکتا در سامانه ثبت‌نام می‌کند. در صورتی که ثبت‌نام موفقیت‌آمیز باشد، کاربر مستقیما وارد سامانه می‌شود. مقدار نام کاربری و رمز عبور بین دو کاراکتر " قرار می‌گیرند. توجه کنید خود " جزو نام کاربری یا رمز عبور نیست و فقط برای جداسازی اجزای مختلف دستور ورودی استفاده می‌شود. همچنین در نام کاربری و رمز عبور کاراکتر فاصله وجود ندارد.

قالب ورودی

```
POST signup ? username "<username>" password "<password>"
```

قالب خروجی

```
OK | Bad Request | Permission Denied
```

خطاها

- در صورتی که نام کاربری تکراری باشد، با خطای Bad Request مواجه می‌شویم.
- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.
- اگر کاربری login کرده باشد و این دستور را وارد کند، خطای Permission Denied چاپ می‌شود.

نمونه ورودی

```
POST signup ? username "pishi" password "meow"
```

نمونه خروجی

```
OK
```

[6] Space

ورود به برنامه

اگر کاربری قبلا در سامانه ثبت‌نام کرده باشد، پیش از استفاده از امکانات سامانه باید وارد سامانه شود. مقدار نام کاربری و رمز عبور بین دو کاراکتر " قرار می‌گیرند. توجه کنید خود " جزو نام کاربری یا رمز عبور نیست و فقط برای جداسازی اجزای مختلف دستور ورودی استفاده می‌شود. توجه کنید که قبل از وارد شدن به برنامه کاربر تنها قادر به ثبت‌نام است.

قالب ورودی

```
POST login ? username "<username>" password "<password>"
```

قالب خروجی

```
OK | Bad Request | Permission Denied | Not Found
```

خطاها

- اگر نام کاربری در سامانه وجود نداشته باشد، خطای Not Found به کاربر داده می‌شود.
- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.
- اگر کاربر رمز خود را اشتباه وارد کند در حالی که نام کاربری موجود باشد، با خطای Permission Denied رو به رو می‌شود.
- اگر کاربری از قبل وارد سامانه شده باشد، با وارد کردن این دستور خطای Permission Denied دریافت می‌کند.

نمونه ورودی

```
POST login ? username "pishi" password "meow"
```

نمونه خروجی

```
OK
```

خروج از برنامه

شخصی که قبلا در سامانه وارد شده بود با وارد کردن این دستور از سامانه خارج می‌شود.

قالب ورودی
`POST logout ?`
قالب خروجی
`OK` | `Permission Denied` | `Bad Request`

خطاها

- در صورتی که کاربری وارد سامانه نشده باشد و این دستور وارد شود، خطای Permission Denied نمایش داده خواهد شد.
- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.

اضافه کردن رویداد

رویداد معمولی

کاربر می‌تواند یک رویداد معمولی را به برنامه‌های خودش اضافه کند. این نوع رویداد یک بازه زمانی مشخص دارد و فعالیت مشخص شده حتما باید در آن بازه زمانی انجام شود. کاربر می‌تواند با ورودی‌های مختلفی که به این دستور می‌دهد، تاریخ، ساعت شروع رویداد و مدت زمان آن را مشخص کند.

هر رویداد معمولی دارای یک شناسه یکتا است که به طور خودکار توسط سامانه اختصاص داده می‌شود که از ۱ برای هر کاربر شروع می‌شود و برای هر رویداد معمولی بعدی یکی زیاد می‌شود.[7]

اگر این رویداد با رویدادهای معمولی یا دوره‌ای دیگر این کاربر تداخل داشته باشد، پیام Overlap چاپ می‌شود و رویداد به برنامه زمانی کاربر اضافه نمی‌شود.

در صورتی که این رویداد با تعطیلات رسمی تداخل داشته باشد، پیام Holiday Found چاپ شود و رویداد به برنامه زمانی کاربر اضافه نمی‌شود. اگر رویدادی با رویداد دیگری تداخل داشته باشد دیگر تداخل رویداد با تعطیلات بررسی نمی‌شود و تنها پیام Overlap چاپ می‌شود.

به جز آرگومان description، سایر بخش‌ها برای این دستور ضروری هستند. توضیحات و عنوان رویداد بین دو کاراکتر " قرار می‌گیرند. استفاده از کاراکتر فاصله[8] در description و title مجاز است.

تضمین می‌شود مدت زمان رویداد (duration) به گونه‌ای ورودی داده شود که رویداد فقط در یک روز باشد.

---

[7] برای توضیحات بیشتر به این بخش مراجعه کنید.

[8] Space

قالب ورودی

```
POST event ? date <YYYY/MM/DD> start_time <HH> duration <HH> title "<title>" description "<description>"
```

قالب خروجی

```
OK | Permission Denied | Bad Request | Overlap | Holiday Found
```

خطاها

- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.
- در صورتی که مقدار آرگومان مدت زمان صفر یا منفی وارد شود، خطای Bad Request چاپ می‌شود.
- در صورتی که هیچ کاربری login نکرده باشد، با وارد کردن این دستور خطای Permission Denied چاپ می‌شود.

نمونه ورودی ۱

```
POST event ? date 1404/12/02 start_time 9 duration 1 title "Quiz" description "DB"
```

نمونه خروجی ۱

```
OK
```

نمونه ورودی ۲

```
POST event ? date 1404/01/13 start_time 11 duration 4 title "DB"
```

نمونه خروجی ۲

```
Holiday Found
```

رویداد دوره‌ای

کاربر می‌تواند یک رویداد دوره‌ای را به برنامه‌های خودش اضافه کند. این نوع رویداد یک بازه زمانی مشخص دارد (شامل تاریخ شروع و پایان). هر رویداد دوره‌ای دارای یک شناسه یکتا است که به طور خودکار توسط سامانه اختصاص داده می‌شود که از ۱ برای هر کاربر شروع می‌شود و برای هر رویداد دوره‌ای بعدی یکی اضافه می‌شود.[9]

---

[9] برای توضیحات بیشتر به این بخش مراجعه کنید

نحوه تکرار این رویدادها به یکی از حالات زیر است:

- روزانه (Daily): رویداد هر روز سر ساعت مشخص شده می‌بایست تکرار شود.
- هفتگی (Weekly): رویداد هر هفته در روزها و ساعت مشخص شده می‌بایست تکرار شود.
- ماهانه (Monthly): رویداد در روز و ساعت مشخص شده در هر ماه می‌بایست تکرار شود.

در صورتی که این رویداد دوره‌ای با رویداد معمولی یا رویداد دوره‌ای دیگری تداخل داشته باشد، این رویداد به برنامه زمانی کاربر در **هیچ روزی** اضافه نمی‌شود و پیام Overlap چاپ می‌شود.

در صورتی که این رویداد دوره‌ای با تعطیلات رسمی تداخل داشته باشد، پیام Holiday Found چاپ می‌شود و این رویداد به برنامه زمانی کاربر در روزهای تعطیل رسمی اضافه نمی‌شود و به سایر روزها اضافه **می‌شود**.

اگر رویدادی با رویداد دیگری تداخل داشته باشد دیگر تداخل رویداد با تعطیلات بررسی نمی‌شود و تنها پیام Overlap چاپ می‌شود.

تضمین می‌شود آرگومان مدت زمان عددی صحیح وارد می‌شود و اعشار ندارد.

خطاهای مشترک

- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.
- در صورتی که مقدار آرگومان مدت زمان صفر یا منفی وارد شود، خطای Bad Request چاپ می‌شود.
- در صورتی که مقدار آرگومان type کلمه‌ای به جز موارد ذکر شده باشند، خطای Bad Request چاپ می‌شود.
- در صورتی که هیچ کاربری login نکرده باشد، با وارد کردن این دستور خطای Permission Denied چاپ می‌شود.

روزانه

در این دستور تمام آرگومان‌ها به جز توضیحات اجباری هستند.

قالب ورودی رویداد روزانه

```
POST periodic_event ? start_date <YYYY/MM/DD> end_date
<YYYY/MM/DD> start_time <HH> duration <HH> type Daily title
"<title>" description "<description>"
```

قالب خروجی رویداد روزانه

```
OK | Permission Denied | Bad Request | Overlap | Holiday Found
```

نمونه ورودی رویداد روزانه

```
POST periodic_event ? start_date 1405/10/29 end_date 1405/11/05
start_time 4 duration 1 type Daily title "Class" description "AP"
```

نمونه خروجی رویداد روزانه

```
OK
```

توضیح نمونه: رویداد مشخص شده به تمام روزها از تاریخ ۲۹ دی ۱۴۰۵ تا تاریخ ۵ بهمن ۱۴۰۵ اضافه می‌شود. از آنجایی که در این بازه تعطیل رسمی وجود ندارد، خروجی پیام OK خواهد بود.

هفتگی

در این دستور تمام آرگومان‌ها به جز توضیحات اجباری هستند. برای آرگومان روزهای هفته، حداقل باید یک روز به عنوان ورودی تعیین شود. روزهای هفته با کاراکتر کاما (,) از یکدیگر جدا می‌شوند. تضمین می‌شود تاریخ شروع و پایان ورودی از نظر روز هفته با مقادیر week_days مطابقت داشته باشند.

قالب ورودی رویداد هفتگی

```
POST periodic_event ? start_date <YYYY/MM/DD> end_date
<YYYY/MM/DD> start_time <HH> duration <HH> type Weekly week_days
<week_day1>,<week_day2>,... title "<title>" description
"<description>"
```

قالب خروجی رویداد هفتگی

```
OK | Permission Denied | Bad Request | Overlap | Holiday Found
```

نمونه ورودی ۱ رویداد هفتگی

```
POST periodic_event ? start_date 1404/01/03 end_date 1404/01/13
start_time 13 duration 2 type Weekly week_days Sunday,Wednesday
title "Software" description "Class"
```

نمونه خروجی ۱ رویداد هفتگی

```
Holiday Found
```

توضیح نمونه ۱: این رویداد باید به تمام یکشنبه‌ها و چهارشنبه‌ها از تاریخ ۳ فروردین ۱۴۰۴ تا تاریخ ۱۳ فروردین ۱۴۰۴ اضافه شود (باید به تاریخ شروع و پایان آن‌ها هم اضافه شود). از آنجایی که ۱۳ فروردین ۱۴۰۴ تعطیل رسمی و چهارشنبه است، پیام Holiday Found چاپ می‌شود. اما به سایر روزها که تعطیل نیستند، این رویداد اضافه می‌شود.

نمونه ورودی ۲ رویداد هفتگی

```
POST periodic_event ? start_date 1404/01/15 end_date 1404/01/22
start_time 15 duration 1 type Weekly week_days Friday title
"Meeting"
```

نمونه خروجی ۲ رویداد هفتگی

```
OK
```

توضیح نمونه ۲: این رویداد به تمام جمعه‌ها از تاریخ ۱۵ فروردین ۱۴۰۴ تا ۲۲ فروردین ۱۴۰۴ اضافه می‌شود. از آنجایی که در این بازه تعطیل رسمی وجود ندارد، خروجی OK خواهد بود.

خطاهای رویداد هفتگی:

علاوه بر خطاهای مشترک که پیش‌تر گفته شد، این خطاها مخصوص رویداد هفتگی هستند.

- در صورتی که مقادیر آرگومان روزهای هفته کلمه‌ای غیر از کلمات Saturday، Sunday، Monday، Tuesday، Wednesday، Thursday و Friday باشد، خطای Bad Request چاپ می‌شود.
- در صورتی که آرگومان روزهای هفته خالی باشد، با خطای Bad Request مواجه می‌شویم.

ماهانه

در این دستور تمام آرگومان‌ها به جز توضیحات اجباری هستند. این دستور یک آرگومان برای روز نیز می‌گیرد که نشان‌دهنده روزی از ماه است که رویداد باید انجام شود؛ تضمین می‌شود مقدار day بین ۱ تا ۳۰ است.

قالب ورودی رویداد ماهانه

```
POST periodic_event ? start_date <YYYY/MM/DD> end_date
<YYYY/MM/DD> day <day> start_time <HH> duration <HH> type Monthly
title "<title>" description "<description>"
```

قالب خروجی رویداد ماهانه

```
OK | Permission Denied | Bad Request | Overlap | Holiday Found
```

نمونه ورودی رویداد ماهانه

```
POST periodic_event ? start_date 1406/04/01 end_date 1406/12/01
day 5 start_time 10 duration 2 type Monthly title "Checkup"
```

نمونه خروجی رویداد ماهانه

```
OK
```

توضیح نمونه: این رویداد به پنجم ماه‌های تیر، مرداد، شهریور، مهر، آبان، آذر، دی، بهمن سال ۱۴۰۶ اضافه می‌شود. از آنجایی که در این روزها تعطیل رسمی وجود ندارد، خروجی OK خواهد بود.

اضافه کردن تسک

تسک‌ها کارهایی هستند که باید تا زمانی مشخص انجام شوند. مانند تکالیفی که در دانشگاه داریم. در این قسمت شما یک تسک جدید به برنامه خود اضافه می‌کنید. تداخل تسک‌ها با رویدادها یا تسک‌های دیگر نباید بررسی شود. هر تسک یک شناسه یکتا دارد که به طور خودکار توسط سیستم به آن اختصاص داده می‌شود. شناسه تسک برای هر کاربر از ۱ شروع می‌شود.

به جز توضیحات، سایر بخش‌ها برای این دستور ضروری هستند. توضیحات و عنوان تسک بین دو کاراکتر " قرار می‌گیرند.

قالب ورودی
`POST task ? date <YYYY/MM/DD> time <HH> title "<title>" description "<description>"`
قالب خروجی
`OK | Permission Denied | Bad Request`

خطاها

- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.
- در صورتی که هیچ کاربری login نکرده باشد، با وارد کردن این دستور خطای Permission Denied چاپ می‌شود.

نمونه ورودی
`POST task ? date 1404/04/04 time 15 title "Boo" description "Doo"`
نمونه خروجی
`OK`

حذف تسک

با این دستور شما می‌توانید یک تسک خاص را از برنامه زمانی خود حذف کنید. ورودی شناسه تسک برای این دستور الزامی است. با حذف تسک شناسه آن در سیستم حذف می‌شود و نمی‌توان آن شناسه را به تسک دیگری اختصاص داد.

قالب ورودی
`DELETE task ? task_id <task_id>`
قالب خروجی
`OK | Permission Denied | Bad Request | Not Found`

خطاها

- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.
- اگر تسکی با شناسه وارد شده در بین تسک‌های کاربر وجود نداشته باشد، با خطای Not Found مواجه می‌شویم.
- در صورتی که هیچ کاربری login نکرده باشد، با وارد کردن این دستور خطای Permission Denied چاپ می‌شود.

نمونه ورودی
`DELETE task ? task_id 2`
نمونه خروجی
`Not Found`

ویرایش تسک

شما با این دستور می‌توانید تسک موجود را ویرایش کنید. برای ویرایش، کاربر باید شناسه تسک و مقادیر جدید برای ویژگی‌های تسک را وارد کند. ویژگی‌هایی که می‌توانند تغییر کنند شامل عنوان، زمان، توضیحات و تاریخ هستند. دقت کنید بخش شناسه تسک ضروری است و کاربر باید حداقل یک ویژگی را تغییر دهد. توجه کنید تنها کلید واژه ویژگی‌هایی که قصد تغییر آن‌ها را داریم، در دستور وارد می‌شوند.

قالب ورودی

```
PUT task ? task_id <task_id> date <new YYYY/MM/DD> time <new HH>
title "<new title>" description "<new description>"
```

قالب خروجی

```
OK | Permission Denied | Bad Request | Not Found
```

خطاها

- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.
- اگر تسکی با شناسه وارد شده در بین تسک‌های کاربر وجود نداشته باشد، با خطای Not Found مواجه می‌شویم.
- در صورتی که هیچ کاربری login نکرده باشد، با وارد کردن این دستور خطای Permission Denied چاپ می‌شود.

نمونه ورودی ۱

```
PUT task ? task_id 1 date 1404/03/03 time 4 title "Happy"
```

نمونه خروجی ۱

```
OK
```

نمونه ورودی ۲

```
PUT task ? task_id 2 title "Woow"
```

نمونه خروجی ۲

```
Not Found
```

گزارش

با استفاده از این دستور، لیستی از تمام کارهایی که کاربر در هر روز باید انجام دهد، نمایش داده می‌شود. برای هر کار، باید تاریخ، ساعت، نوع، عنوان و توضیحات آن (در صورت وجود) نمایش داده شود. توجه کنید که این دستور لیستی از کارهای کاربر در هر روز را نمایش می‌دهد؛ لذا برای مثال اگر کاربر یک رویداد روزانه برای یک بازه ۷ روزه دارد باید در گزارش آن بازه به ازای هر روز گزارش آن رویداد جداگانه وجود داشته باشد. اگر در روزی هیچ کاری وجود نداشته باشد، آن روز در گزارش نمی‌آید.

ترتیب نمایش خروجی بر حسب نزدیک‌ترین فعالیت است. در صورتی که انتخاب بیش از یک کار داشتید بر اساس نوع فعالیت به ترتیب رویداد دوره‌ای، رویداد معمولی و تسک اولویت‌بندی کنید. در صورتی که باز هم بیش از یک انتخاب وجود داشته باشد بر اساس شناسه به شکل صعودی اولویت بدهید.

آرگومان to برای این دستور اجباری است اما آرگومان‌های type و from اختیاری هستند.

اگر آرگومان from وارد نشود، باید تمامی کارها از تاریخ مبدا سامانه تا تاریخ مشخص شده چاپ شود.

همچنین این دستور می‌تواند شامل یک آرگومان اختیاری type باشد که فقط باید کارهایی از آن نوع (رویداد معمولی، رویداد دوره‌ای و تسک) را چاپ کند. مقادیر مجاز برای آرگومان type شامل موارد زیر است:

- رویداد معمولی: event
- رویداد دوره‌ای: periodic_event
- تسک: task

تضمین می‌شود تاریخ آرگومان to بعد از تاریخ آرگومان from می‌باشد.

قالب ورودی

```
GET report ? from <YYYY/MM/DD> to <YYYY/MM/DD> type <type>
```

قالب خروجی

```
task "<title>" on <YYYY/MM/DD> at <time>: "<description>" |
event "<title>" on <YYYY/MM/DD> from <start_time> for <duration>
hours: "description" |
periodic_event "<title>" on <YYYY/MM/DD> from <start_time> for
<duration> hours <type>: "<description>" |
Permission Denied | Bad Request | Empty
```

خطاها

- اگر کاری وجود نداشت باید خطای Empty چاپ می‌شود.
- در صورتی که مقدار آرگومان نوع کلمه‌ای غیر از کلمات مجاز وارد شود، خطای Bad Request چاپ می‌شود.
- در صورتی که هیچ کاربری login نکرده باشد، با وارد کردن این دستور خطای Permission Denied چاپ می‌شود.
- در صورتی که اجزای دستور به درستی وارد نشوند، با خطای Bad Request مواجه می‌شویم.

نمونه ورودی ۱

```
GET report ? to 1406/12/30
```

نمونه خروجی ۱

```
periodic_event "Software" on 1404/01/03 from 13 for 2 hours
Weekly: "Class"
periodic_event "Software" on 1404/01/06 from 13 for 2 hours
Weekly: "Class"
periodic_event "Software" on 1404/01/10 from 13 for 2 hours
Weekly: "Class"
periodic_event "Meeting" on 1404/01/15 from 15 for 1 hours Weekly:
periodic_event "Meeting" on 1404/01/22 from 15 for 1 hours Weekly:
task "Happy" on 1404/03/03 at 4: "Doo"
event "Quiz" on 1404/12/02 from 9 for 1 hours: "DB"
periodic_event "Class" on 1405/10/29 from 4 for 1 hours Daily:
"AP"
periodic_event "Class" on 1405/10/30 from 4 for 1 hours Daily:
"AP"
periodic_event "Class" on 1405/11/01 from 4 for 1 hours Daily:
"AP"
periodic_event "Class" on 1405/11/02 from 4 for 1 hours Daily:
"AP"
periodic_event "Class" on 1405/11/03 from 4 for 1 hours Daily:
"AP"
periodic_event "Class" on 1405/11/04 from 4 for 1 hours Daily:
"AP"
periodic_event "Class" on 1405/11/05 from 4 for 1 hours Daily:
"AP"
periodic_event "Checkup" on 1406/04/05 from 10 for 2 hours
Monthly:
```

نمونه ورودی ۱

```
periodic_event "Checkup" on 1406/05/05 from 10 for 2 hours
Monthly:
periodic_event "Checkup" on 1406/06/05 from 10 for 2 hours
Monthly:
periodic_event "Checkup" on 1406/07/05 from 10 for 2 hours
Monthly:
periodic_event "Checkup" on 1406/08/05 from 10 for 2 hours
Monthly:
periodic_event "Checkup" on 1406/09/05 from 10 for 2 hours
Monthly:
periodic_event "Checkup" on 1406/10/05 from 10 for 2 hours
Monthly:
periodic_event "Checkup" on 1406/11/05 from 10 for 2 hours
Monthly:
```

نمونه ورودی ۲

```
GET report ? from 1407/01/01 to 1407/01/02 type task
```

نمونه خروجی ۲

```
Empty
```

ساخت شناسه

همان‌طور که در توضیحات دستورات برنامه ذکر شد، موجودیت‌های برنامه دارای شناسه هستند. این شناسه‌ها به شکل صعودی با توجه به سطح هر موجود تعیین می‌شوند. سطح موجود می‌تواند مختص به کاربر یا کل برنامه باشد. رویدادهای دوره‌ای و معمولی و تسک همه در سطح کاربر شماره‌گذاری می‌شوند.

سطح کاربر

در این سطح شناسه اختصاص داده شده به موجود، برای کاربر یکتا است. به این معنا که برای هر کاربر از ۱ شروع می‌شود و به شکل صعودی، یکی یکی اضافه می‌شود. بنابراین هر کاربری می‌تواند تسک با شناسه ۱ داشته باشد.

مثال

1. کاربر شماره ۱ یک تسک ایجاد می‌کند. شناسه آن برابر ۱ است.
2. کاربر شماره ۲ یک تسک ایجاد می‌کند. شناسه آن برابر ۱ است.
3. کاربر شماره ۱ یک تسک دیگر اضافه می‌کند. شناسه آن برابر ۲ است.

سطح برنامه

در این سطح شناسه اختصاص داده شده به موجود، در کل برنامه یکتا است. به این معنا که در ابتدای اجرای برنامه از ۱ شروع می‌شود و یکی یکی به شکل صعودی اضافه می‌شود.

مثال

1. کاربر شماره ۱ یک موجود از این دسته ایجاد می‌کند. شناسه آن برابر ۱ است.
2. کاربر شماره ۲ یک موجود از این دسته ایجاد می‌کند. شناسه آن برابر ۲ است.
3. کاربر شماره ۱ یک موجود از این دسته ایجاد می‌کند. شناسه آن برابر ۳ است.

توجه کنید این نوع موجودیت در این فاز وجود **ندارد** و در فاز **آینده** به آن می پردازیم.

نکات و نحوه تحویل

- تحویل این تمرین در سامانه گیت‌هاب انجام می‌شود. برای انجام تمرین لطفا از طریق این لینک وارد شوید، پس از آن باید شماره دانشجویی خود را انتخاب کنید (دقت کنید که با کمک این شماره دانشجویی به شما نمره خواهیم داد، لطفا در انتخاب درست شماره دانشجویی حتما دقت کنید، در صورتی که به مشکل خوردید با دستیاران در ارتباط باشید). پس از آن به صفحه‌ای منتقل می‌شوید که در آنجا می‌توانید تمرین جدید را قبول کنید، پس از قبول کردن تمرین یک مخزن[10] در AP Assignments برای شما ساخته می‌شود و باید کدهای خود را در آنجا قرار دهید.
- پس از انجام تمرین و بارگذاری در گیت‌هاب، کد Hash آخرین کامیت[11] را به همراه شماره دانشجویی خود در سامانه ایلرن آپلود کنید (در خط اول شماره دانشجویی، پس از آن از Enter استفاده کنید و به خط بعد بروید و پس از آن Hash آخرین کامیت). نمونه متن خواسته شده در سامانه ایلرن (بخش <last_commit_hash> و <sid> را جایگزین کنید):

```
<sid>
<last_commit_hash>
```

نمونه:

```
810100000
bad8fbcdcfa3b9feb371a31e0c370150aa870b18
```

- دقت کنید که عدم رعایت ساختار گفته شده در آپلود یا تغییر ساختار فایل‌ها در مخزن (می‌توانید به دلخواه خود فایل اضافه کنید و ... اما اسم و ساختار فایل‌هایی که در ابتدا به شما داده می‌شود نباید تغییر کند) باعث کسر 5 درصد از نمره شما خواهد شد.
- دقت کنید که فایل makefile باید در صفحه اول مخزن باشد و در پوشه‌ای قرار نداشته باشد و در آن مشخص کنید که از استاندارد c++20 استفاده می‌کنید.
- نام برنامه قابل اجرای شما باید UTrello (بدون هیچ پسوندی مانند exe یا out) باشد و پس از ساخته شدن در کنار makefile قرار بگیرد (داخل پوشه‌ای فایل خروجی ساخته شده را قرار ندهید).
- درستی برنامه شما از طریق آزمون‌های خودکار سنجیده می‌شود؛ بنابراین پیشنهاد می‌شود که با استفاده از ابزارهایی مانند diff خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- با توجه به حجم نسبتا زیاد این فاز از تمرین توصیه می‌شود قبل از پیاده‌سازی کد طراحی اولیه‌ای برای منطق برنامه و روندهای آن مثل ثبت‌نام و ... انجام دهید و پس از این طراحی شروع به پیاده‌سازی آن کنید. از آن جایی که در فازهای بعدی شما باید رابط کاربری برنامه‌ی خود را از

---

[10] Repository

[11] Commit

command-line به روش‌هایی دیگر تغییر دهید، بهتر است تا طراحی برنامه‌ی شما طوری باشد که کمترین وابستگی میان منطق برنامه و رابط کاربری آن وجود داشته باشد.

- برنامه شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++20 ترجمه و در زمان معقول برای ورودی های آزمون اجرا شود.
- درستی برنامه شما از طریق آزمون‌های خودکار سنجیده می‌شود؛ بنابراین از درستی کامل قالب خروجی برنامه خود اطمینان حاصل کنید و از دادن خروجی‌هایی که در صورت پروژه ذکر نشده است اجتناب کنید.
- سوالات خود را تا حد ممکن در فروم درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن‌ها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این فاز پروژه ارتباط برقرار کنید.
- توجه داشته باشید که حالت‌های خاصی که در صورت پروژه ذکر نشده است در تست‌های خودکار نخواهد بود و می‌توانید به هر شکلی که مد نظر دارید آن‌ها را مدیریت کنید.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

نمرات

- تمیزی کد
  - رعایت کردن نام‌گذاری صحیح و انسجام[12]
  - عدم وجود کد تکراری
  - رعایت دندانه‌گذاری[13]
  - عدم استفاده از متغیرهای گلوبال
  - استفاده **صحیح** از متغیرهای ثابت[14] به جای Magic Value-ها
  - ساختاردهی کد در قالب توابع کوتاه که فقط یک کار را انجام می‌دهند
- درستی کد
  - آزمون‌های خودکار
  - پیاده‌سازی صحیح کارکردهای خواسته شده
- طراحی
  - طراحی صحیح و منطقی در شی‌گرایی و ارث‌بری
  - رعایت Encapsulation

[12] Consistency

[13] Indentation

[14] Constant

- جداسازی منطق کد از ورودی/خروجی و استفاده از کلاس جداگانه برای مدیریت دستورات
- استفاده مناسب از استثناها برای مدیریت خطا
- میک‌فایل و چندفایلی

دقت کنید که موارد ذکر شده لزوما کل نمره شما را تشکیل نمی‌دهند و ممکن است با تغییراتی همراه باشند.